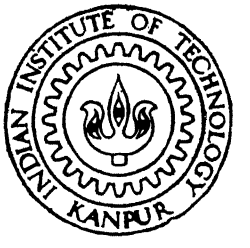
System Design Considerations in Some Geographical Information Systems

By

Nirupma Kulshreshtha



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

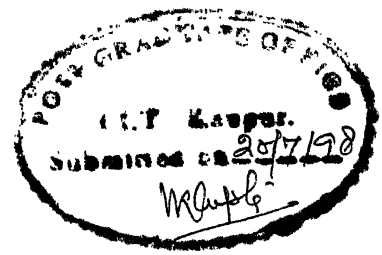
JULY, 1998

System Design Considerations in some Geographical Information Systems

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by
Nirupma Kulshreshtha

to the
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
July, 1998



Certificate

Certified that the work contained in the thesis entitled "*System Design Considerations in some Geographical Information Systems*", by Ms.*Nirupma Kulshreshtha*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Prof. K. R. Srivathsan)	(Prof. T. V. Prabhakar)
Professor,	Professor,
Electrical Engineering,	Computer Science and Engineering,
IIT Kanpur.	IIT Kanpur.

July, 1998

To my father,

Abstract

Geographical Information Systems (GIS) are rapidly proliferating and increasing in importance in diverse application areas such as emergency vehicle routing, mapping of census data and location of housing, industry etc. Not much effort has gone into the standardisation in the data structures or tools associated with GIS. The work of designers and end users of GIS is therefore made difficult. In this thesis, a brief review of some recent efforts in the understanding of relations across the DBMS associated with GIS is given. The models and attributes of typical GUI for GIS proposed in literature are also reviewed. Based on these, some system design considerations for GIS are proposed. A sample skeletal implementation of a GIS for network monitoring has been implemented over a JAVA platform.

Acknowledgment

I express my sincere gratitude to my thesis supervisor Prof. K. R. Srivathsan for his constant guidance and valuable suggestions that he gave me throughout this work.

I am indebted to Prof. T. V. Prabhakar for encouraging and supporting me in every possible way during my Mtech programme, specially during the time of submission. Successful completion of this thesis would have never been possible without his help.

I am sincerely thankful to my friend Shalini, who cared for me during my stay in this campus. I am also thankful to all those who have helped me during my stay in Kanpur.

Special thanks to my mother, brother and sister-in-law for their love and affection I have been receiving.

Nirupma Kulshreshtha

Contents

List of Figures

Chapter 1

Introduction

An efficient human-computer interaction[1] is a task of increasing importance in many new database applications such as CAD/CAM[1], environmental management[2], multimedia[3] and geographic applications[4].

Geographic applications[4] may vary widely but they all have common aspects due to the spatial component[5] of their data. The conceptual problems encountered in designing Geographical Information Systems are partly due to the merger of two independent fields, geographic DBMSs on the one hand, and graphical user interfaces on the other hand. Although these areas have evolved considerably during the past ten years, only little effort has been made to understand the problems of connecting them in order to efficiently manipulate geographic data on a display. We focus here on Geographical Information System (GIS) using a database management system (DBMS) as a support.

Although DBMSs sometimes provide basic features for handling geographic data like storing large data in the database, searching the database etc., they are usually not designed to manipulate them graphically in an efficient way. In addition, only a few of them offer a customizable and extensible graphical user interface (GUI).

GIS is a software system which joins database management and mapping capabilities within a single computer package. It thereby allows the storage, analysis, and presentation of all information (geographic and other) relevant to a project. The GIS mapping component facilitates the easy interpretation of database information

as by clicking on any point on a map it will bring up the relevent part of a database on the computer screen. Alternatively the result of a query into the database can be presented as a map. The maps and associated information can be presented on a computer screen or printed out and disseminated to users. A GIS can be used in many scientific and management disciplines as an information system to address[2] a broad range of environmental management problems like map visualization and querying.

Map visualization and querying require an elaborate graphical user interface together with a powerful querying technique that conventional DBMS environments are not usually designed for. In case of distributed DBMS, individual site mantains its own database. Hence global relational qeries can be asked by the users from a network of locally managed databases.

User interface are a fundamental component of applications using as support Geographical Information System (GIS) and Geographic Database Management Systems (DBMS). Maps are the basic objects handled by such DBMS. There are two classes of functions in Geographical Information Systems (GIS):-

- the display of maps and,
- interactive queries on these maps.

It is clear that these user requirements cannot be handled by regular interface tools for database systems. In particular, GIS's present specific features such as the following:-

- Visual mapping of the data supplied by the GPS.
- Graphic User-interface to enter the details of each node in the map.
- Storage of this information corresponding to each map in the separate database.
- Creation of map from the information stored in the database.
- Search engine to extract the information from the database.

GIS are concerned with many different fields, such as planning, resource management, traffic control, etc., which may have an impact on the functionalities of a GUI. The challanges in designing general GISs are partly due to this diversity and to the specific characteristics of GIS end-users. Firstly, they are usually naive users who are unable to formulate complicated queries. Asking them to master database aspects such as schemas is therefore hardly realistic. Secondly, they sometimes only have a vague idea of what they are looking for and rather than expressing exact queries, they may prefer to browse geographic information.

1.1 Scope

The goal of this thesis is to investigate the problem of designing a Geographical Information System (GIS), and to propose a toolkit-based solution that satisfies a large number of users.

In this thesis, we firstly study the major aspects of visualizing and querying geographic information within a database management system (DBMS) and their impact on the design of Geographical Information System (GIS). We then present a map creating, editing and querying model as well as a general GIS architecture. We also discuss a GIS application built on this architecture.

1.2 Organization

This thesis is organized as follows. In Chapter 2, we explain the various GISs studied. In Chapter 3, we give the overview of GPSR 2000. In Chapter 4, we propose a design for GIS, which is a coupling of user interface with a distributed geographic DBMS. In Chapter 5, we discuss the implementation details of our GIS. Finally, Chapter 6 draws some conclusions.

Chapter 2

A Brief Review of GIS

2.1 Introduction

In this chapter, we'll survey few GIS models to gain some insight into the functionalities that a GIS should provide. The next section introduce the concept of GIS in detail and briefly explain three GIS models - Mapgets[6], GODOT[7] and GISER[8]. The last section presents the conclusions.

2.2 Geographical Information Systems

Geographical Information System (GIS) is a computer system capable of assembling, storing, manipulating, and displaying geographically referenced information.

GIS provides the facility to extract the different sets of information from a map (roads, settlements, vegetation, etc.) and use these as required. This provides great flexibility, allowing a paper map to be quickly produced which exactly meets the needs of the user. However, GIS goes further, because the data are stored on a computer, analysis and modelling become possible. For example, one might point at two buildings, ask the computer to describe each from an attached database and then to calculate the best route between these.

A GIS makes it possible to link, or integrate, information that is difficult to associate through any other means. Thus, it can use combinations of mapped variables

to build and analyze new variables.

A critical component of a GIS is its ability to produce graphics on the screen or on paper that convey the results of analysis to the people who make decisions about resources. Wall maps and other graphics can be generated, allowing the viewer to visualize and thereby understand the results of analyses or simulations of potential events.

The GIS data models can be categorized into *field-based* models and *object-based* models. *Field-based* model see the world as a continuous surface (layer) over which features vary (e.g., elevation). Layer algebra[9] provides a field-based view. It defines a set of operations which can manipulate different layers to produce new layers. The *object-based* model treats the world as a surface littered with recognizable object, which exist independent of their locations. GeO_2[10], Mapgets[6], GraphDB[11], GODOT[7], Worboy[12] and OGIS[13] are some attempts to model GISs using the object-based approach. The GERM[4] model attempts to unify the two approaches and provides a set of concepts as an add-on to the ER model for modeling GISs. In this chapter, we'll study few of these models in detail.

2.2.1 MAPGETS

Architecture

Mapget stands for map wid*get*. It includes a support level (low level graphic tools facilities and DBMS functions) and a Graphical Database User Interface (GDUI) level, which represents the kernel. At the supprot level, it uses on the one hand the O_2 DBMS for storing and querying the map database and on the other hand, X Windows (version X11 R4)[14] and OSF/Motif[15] for coding the interface. The functional architecture of the kernel is displayed in Figure 1.

The GDUI is composed of three modules:-

1: Mapget: This is used to display maps in windows.

2. Main: "Main" deals with images and it communicates both with the mapget module and with the connection module.

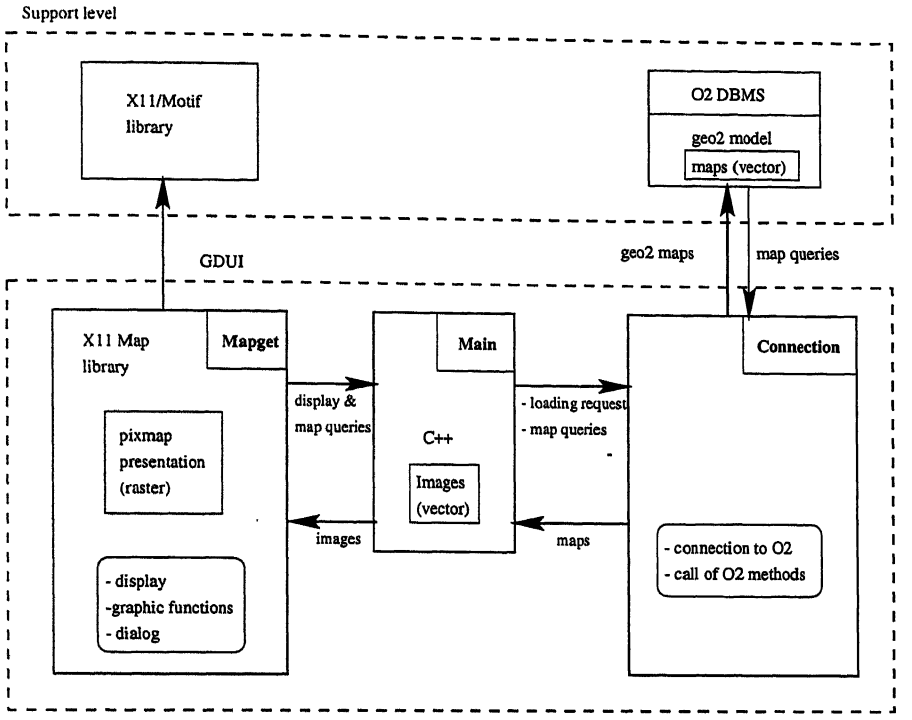


Figure 1: Functional Architecture of GDUI

3. Connection: This module is the interface with the database. Main module interacts with this module for map display and queries.

When a map (stored in the database) is to be displayed, it has first to be converted into an image. This is performed by the connection module, which knows both data models. This image is handled by the main module, kernel of the GDUI. The connection module, which is the interface between the DBMS and the main module, is in particular in charge of handling all exchanges with the DBMS when displaying and querying. The connection module can be used for other applications (e.g., applications using statistics), in order to make the bridge between an external module and a DBMS.

In general, the connection module should support two functionalities:-

1. Communication with the DBMS (queries, data flows).

2. Mapping from database objects onto external (GUI) objects.

Map Editing and Querying Features of Mapgets

Maps are collection of geometric objects. A geometric object has usually two parts, an alphanumeric one, or description, and a spatial one corresponding to its geometry. In addition, a geographic object can be atomic or complex i.e. made of other geographic objects. A map is more than a complex geographic object since it includes information on data it contains such as the coordinates system, the source of information, etc. A geographic database is a collection of maps.

Map editing is concerned with two major aspects:

1. displaying and

2. modifying maps.

Displaying maps usually refers first to the display of the spatial part of their geographic objects. The corresponding displayed objects are called cartographic objects. The geographic objects' description can be displayed after pointing to cartographic objects on the screen.

The main feature of modification process is the possibility of modifying objects interactively and storing the modifications in the database. For example, the user may erase the boundary between two countries (at a lower level, between two polygons). Since objects of a displayed map can change after an interaction with the end user, a map itself can be modified. The interface, which is between the user and the database, must send this kind of information to the database which will update the modified map.

As far as map querying is concerned, the problem is more complex than in the case of "standard" queries (purely alphanumeric) since map queries sometimes need graphic parameters. As a consequences, a specific interaction with the end-user has to be considered.

Map Display: Basic Functionalities and Visual Aspects

Displaying maps includes the following basic functionalities:

1: Display data of type raster (bitmap) or vector (point, line, polygon).

2. Display alphanumeric data (description of a geographic object).

The geometry of a map can be either in a vector format or in a raster format (satellite images). Mapgets focus on maps whose geometry is stored in a vector format. Each geometric object stored in the database has alphanumeric description associated with it. Alphanumeric data is stored in the database independently of maps. This data is fetched from the database at the time of display.

For a minimum of map understanding and map manipulation, the following is required:

1. Overlay vector maps.

2. Allow one to change scale (e.g., zoom) on a whole map or on a subset of a map.

3. Attach a legend(color, pattern, text, symbol) to a map or to a subset of a map.

4. Display data with various scale and different representations (mix of the two previous points). A well-known related problem is the problem of generalization. For instance, it allows one to modify the representation of geographic data according to the scale (e.g., for displaying them in a smaller space). This means that new cartographic objects have to be computed. To give examples, the four following points may improve the clarity of a map display:

   - Give a new representation to a geographic object according to the scale. In the worst case, if the scale is too small, the geographic object will disappear unless it has a "semantic importance"[16].
   - Simplify the contour of a 2-dimensional object (remove points on the border), or of a 1-dimensional object (drop points on a line).
   - Simplify the display of geographic objects by using the concept of aggregation when there exist subobjects that represent a partition of the plane.
   - Simplify the display of a complex geographic object. For instance, on a map show few geographic objects, instead of showing all geographic objects.

▮ *sub Windows and Interaction with End-Users*

Maps are displayed in windows with which users have a particular interaction. This includes:

- Map display (geometry):
- Display several maps in a same window without overlaying them.
- Possibly show a restricted part of a map (the rest can be seen by scrolling).
- Overlay several views of the same geographic space in a same window (e.g., states, cities etc.).

Although the first two tasks seem to be GUI issues rather than GIS isues, but the later two need to be handled in a specific way in geographic applications.

- Interaction with the user on the display:
  - Allow an end-user to select objects on the screen.
  - Provide the possibility to draw areas on the screen (e.g. a rectangular for a zoom)
  - Give the user the possibility to modify objects interactively (e.g., erase a boundary between two polygons).
  - Provide query language facilities, either graphically such as SQL extended to spatial concepts[17]. Such a language has a tight connection with the GUI because of the interactive nature of queries in geographic databases. In an object-oriented environment, some queries corresponding to operations on maps such as clipping, windowing, map overlay, etc. can be defined as methods invoked from a GUI.
- At a meta level, provide (graphic) tools for browsing the database and the map library, for instance to select appropriate maps for a given study.

Interaction with a Database

Given the previous requirements, it is clear that the interface has to communicate with the database for at least:

- Getting the objects to be displayed: Maps as a whole, i.e., the geometric part of their geographic objects but also their description.
- Attaching a representation such as color to a geographic object, depending on the information it contains. This is usually done by reading a legend stored in a database independently of maps. Then the corresponding cartographic objects are created.
- Updating objects i.e., changing the value of a geographic object in the database.
- Querying maps: 1)Alphanumerical querying and 2)Graphical querying.

The main communication functions between the user interface and the database are given below[1].

- Display a map:
  `ui:MapDisplay(map).`
- Get a geographic object (go) in a (database) map from the coordinates of a cartographic object (co):
  `connection:GetObject(map, co) -> go.`
  /* Assuming there may be several maps displayed on the screen */
- Update a geographic object in a (database) map
  `connection:UpdateObject(map,go).`
- Apply a (db) function f on a map:
  `db:f(map, args) -> result`, where `args` is a list of either alphanumerical or graphical arguments and `result` the result of a query.

[1]Regarding the notation, in order to remain general we refer to the alphanumeric part of a geographic object as "description". Functions that refer to database operations are prefixed by `db:`, while user interface functionsare prefixed by `ui:` and functions that are the link between the interface and the database are prefixed by `connection:`.

Connecting an External Module to a DBMS

The user interface of a geographic application can be considered as a kernel independent of a DBMS although it has to be connected to it. This holds for many specific applications (multimedia applications such as hypertext, statistics with persistent data, etc.) whose particularity is to perform treatments that are usually out of the scope of the DBMS.

When an external module is used together with a DBMS, one question that arises concerns the division of tasks among the DBMS and the external module. The functionalities of a DBMS are limited and it is sometimes not clear whether some specific tasks should be performed by it or by an external module.

There exist basically two different ways to define a specific application within a DBMS environment, i.e., to connect an external module to a DBMS. Either the integration is *strong* (no real external module, Figure 2(a)) or it is *weak* (Figure 2(b)). The way from strong to weak integration is gradual. For instance, an in-between solution considers an extensible DBMS (i.e., a DBMS with hooks for extensions at different levels, see[5]), so the DBMS plays the role of an integrator. In the case of strong integration, the potential of the under-lying DBMS is exploited as much as possible, and all concepts are embedded within the same homogeneous environment. In particular, only one data model exists, and if the (virtual) external module needs special structures, these structures will be defined using the database model (e.g., in[3]).

In the case of weak integration (Figure 2(b)), two independent systems coexist: The DBMS on the one hand and the external module on the other hand, and the main problem is to build the connection between them. Another problem concerns the repartition of tasks among the two systems. In the case of GDUIs for example, it is not clear what should be left to the GUI module and what should be defined in the geographic DBMS.

Integrating a Graphical User Interface into a DBMS

Compared to other possible external modules, geographic user interfaces have a tight connection with a DBMS since, among other things, they reflect the contents of the

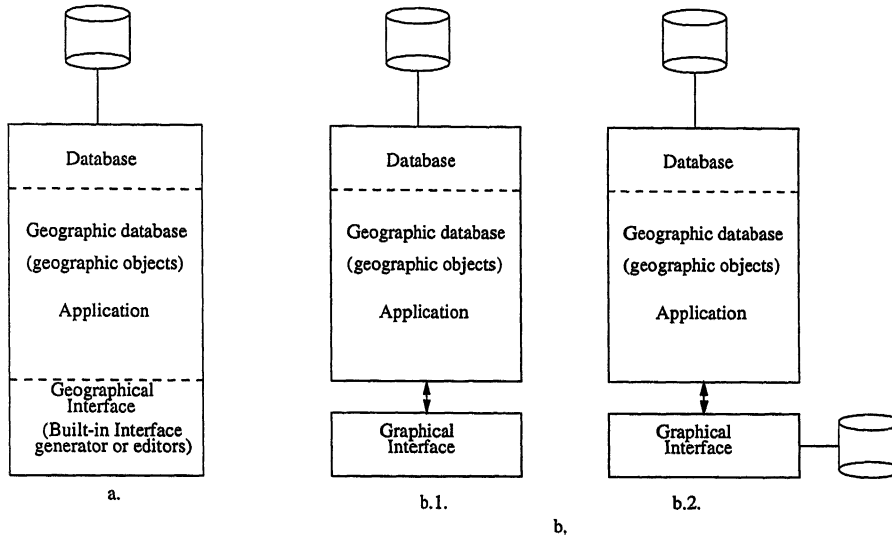


Figure 2: Integrating a user interface with a DBMS

database at a given time and show a view of database objects. A few database environments provide tools, unfortunately not always sufficient, to design a graphic user interface. Then for designing a GDUI, the problem of choosing a type of integration still remains and the two opposite approaches mentioned above can be considered: Either graphic concepts are embedded into the DBMS environment (such as in[2]) (Figure 2(a)), or they are completely independent of (such as in[6], Figure 2(b)). In Figure 2.b.2, the GUI objects are stored in a database, which is not necessarily the one used to store the geographic objects.

Strong Integration into a DBMS

In this approach, (Figure 2(a)), there is no real distinction between geographic objects and cartographic objects since cartographic objects are the (unique) views of geographic objects. In other words, there is only one environment to deal with. There are two cases. In the first case, the DBMS provides an interface generator, which allows one to design editors for new types of data such as geographic data. In the second case, no tools are provided and display routines (map editors) need to be defined in the application.

In case 1 (existence of an interface generator), the advantage is that the developer's task is obviously facilitated. There is no need to consider communication aspects between the interface and the database for getting and updating objects because the framework to do that already exists. In case 2 (no interface generator), the application must include routines for displaying objects. For instance, in the basic GeO_2[10] system, displaying objects is done via methods on geographic objects that invoke Xlib routines.

However, strong integration suffers from major drawbacks. In both previous cases, the DBMS application gives an object the "order" to display itself on the screen, and everything is done within the database environment which is a complete geographic DBMS. It is clear that this *ad hoc* solution is data model dependent. The interface can be considered as an application on top of the DBMS, which leads to limits if the model or the application change. Moreover, it seems difficult to consider many different databases simultaneously, although this is useful in geographic applications when data from different sources are to be merged (say for instance a set of maps stored at a big scale in some DBMS and an another set stored at a small scale in another DBMS). In addition, in geographic applications, it is not unusual to have data stored under different systems (e.g., Unix and MS/DOS).

If no interface generator is provided (Case 2), there is more flexibility in the design either, or the programmer has to master the low levels of the DBMS (with possible use of system primitives) together with graphic routines.

Weak Integration: Separated Conception

In this approach (Figure 2(b)), the user interface is developed separately and is eventually connected to a paticular DBMS, usually in an ad-hoc manner. The user interface can be developed using graphic packages (such as GoPATH[18]. IlogViews[19], etc.).

Compared to the previous approach, the advantages here rely on the fact that the interface is adapted to the needs of geographic applications. From a design point of view, there is a total independence between the database and the interface, and the designer needs not use neither a given a display philosphy nor imposed

graphic tools. The advantage of efficient graphic routines defined in a specialized package can be taken, which leads to a more powerful interface than in the case of strong integration. New tools can easily be added, such as sophisticated editors for statistics.

The main drawback of this approach is that the connection with the database needs to be defined by the developer (e.g., for getting objects, querying, sending data flows). It is now widely recognized that coupling systems is cumbersome for many reasons, from a logical level (type mapping) to a physical level. For instance, if the geographic interface is developed from scratch, a low level communication dialog has to be defined (e.g., using Unix sockets). Moreover, if the DBMS changes, the communication layer has to be reprogrammed. To implement one of the first user interfaces of the Sequoia project[20] built on top of Postgres[21], a *visualization software package* (AVS) was chosen, which required the writing of an AVS-Postgres bridge (obviously very specific). A more modular solution was chosen in the Godot system[7], developed on top of ObjectStore[22] whose GUI uses the GoPATH[18] package together with an "interactive interface" level.

2.2.2 GODOT

Introduction

This section describes the design of an object oriented geographic information system called GODOT (*Geo*graphic *D*ata Management With *O*bject-Oriented *T*echniques). The idea behind GODOT is to develop a GIS prototype on top of a commercial object-oriented database system (OODB) by adding appropriate classes and methods.

Using this approach, GODOT differs from most other recent GIS developments in the following ways:-

1. GODOT's object-oriented data model allows the representation of highly complex geographic information (e.g., in environmental application) as a network of geographic objects.

2. The GODOT data model is *extensible* by user-defined classes and methods.

3. GODOT's underlying OODB is a general purpose system which allows for both spatial data and ordinary tabular data to be seamlessly integrated within the same environment.

4. GODOT is based on a commercial OODB and therefore participates in new developments on the database market. This may concern features such as query language standards, graphical tools, transaction management. and distributed processing.

System Architecture

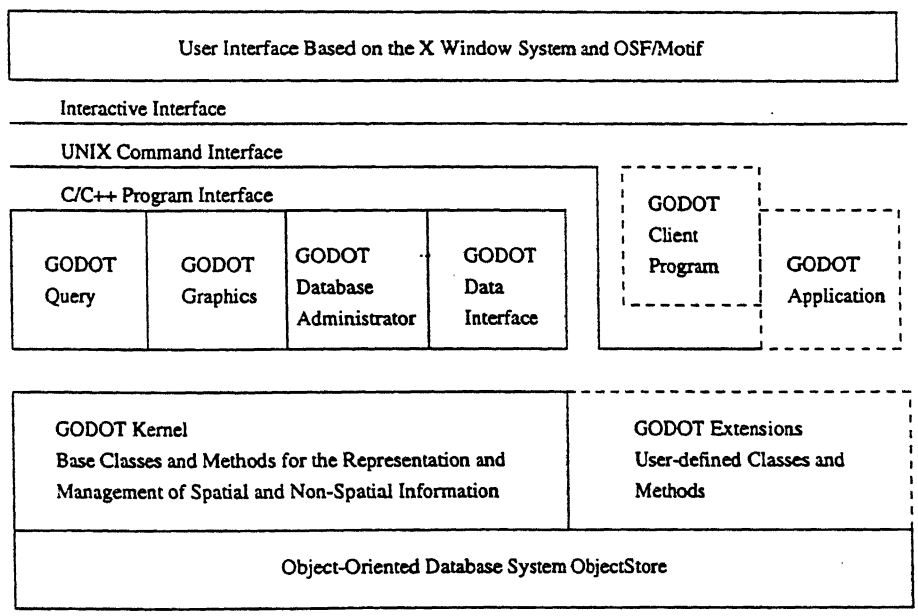


Figure 3: GODOT Architecture

The GODOT system has a four-layer architecture (Figure 3), consisting of:

1. A commercial OODB.

2. An extensible kernel with classes and methods for the representation and management of complex spatial and non-spatial data objects.

3. A collection of base components for query processing, graphics, database administration, and data exchange.

4. Several user interface modules, including a C/C++ program interface, a UNIX command interface and a graphical user interface based on the X Window and OSF/Motif.

OODB and Kernel

The GODOT kernel is implemented directly on top of the commercial OODB ObjectStore[22]. ObjectStore is a fully object-oriented database system in the sense of the OODB Manifesto[23]. This covers in particular the basic database functionalities, including transaction management and indexing. Some of these functionalities have been adapted and extended for a more efficient management of spatial data.

The GODOT kernel contains the definition of classes and methods that are crucial for the representation and management of geographic information. In particular, the implementation of the GODOT data model is located here. Furthermore, the spatial clustering and indexing components will be located in the kernel. Like other GODOT components, the kernel is implemented in the object-oriented programming language C++.

The GODOT data model can be extended by additional classes and methods. This facilitates the custoization of the system to serve any particular application.

Base Components

The *query component* contains several GIS-specific language elements to enhance the query language of ObjectStore. It interacts directly with the interface modules described in the next section. ObjectStore allows the call of a user-defined method within a query; this feature has been used to extend the query language with spatial and topological predicates.

The query component is tightly linked to a graphics component that manages the graphical representation of the geographic information. In particular, it is common to use the graphics component to specify parts of a query by pointing to certain objects on the screen. This includes the ability to update geographic information by manipulating the corresponding graphic objects interactively.

The *database administrator component* provides the usual features for database schema manipulation, user administration, and system support. Some of the ObjectStore's functionalities can be used directly, while others need to be adapted to manage spatial data efficiently.

A major issue in GIS is the exchange of geometric data encoded in different formats. GODOT supports the integration of *data interfaces* as base components; these can be activated through any of the interface modules described in the following section. GODOT also has its own external data format, which is a subset of C++, encoded in ASCII. This external data format can be read easily by other systems. The execution of the code leads directly to the generation of the corresponding object classes and instances in the given database.

User Interfaces

One of GODOT's core functionalities is to be a GIS data server for a diverse and distributed collection of applications. For this purpose, GODOT offers a variety of client-server style interfaces. An *interactive interface* under the X Window System gives high-level graphical access to GODOT, especially for the occasional or non-expert user. A different kind of access is provided by the *UNIX command interface*, where GODOT queries can be formulated by means of specialized commands that form an extension of the UNIX shell. Command procedures can then be implemented as shell scripts. Finally, a *C/C++ program interface* makes the GODOT modules available as a program library. This interface is typically used for more complex GODOT applications; it also allows remote access via remote procedure calls.

Data Model

The GODOT data model is partitioned into three categories of objects:

1. *Thematic objects*: These are used to represent real-world objects. An important subcategory of thematic objects are the *geographic objects or* geo-bjects. A thematic object is a geo-bject if it is geometric in nature, i.e., if it has a spatial extension.

2. *Geometric objects* or *geometries*: These are used to describe the geometric feature of geo-objects.

3. *Graphic objects*: These are used for the display of thematic objects. *Cartographic objects* are an important subcategory of graphic objects.

Thematic Objects and Geo-Objects

In the GODOT data model, geographic and environmental information is represented by so-called *thematic objects*. Thematic objects may be simple or complex, i.e., composed of several other thematic objects. Thematic objects may have different kinds of attributes to represent a variety of geographic and environmental features:

1: Attributes of an elementary type (e.g., text strings or real numbers)

2. Attributes of a complex type (e.g., embedded classes in C++)

3. Attributes of a referential type (e.g., pointers to other thematic objects)

The most important subset of the thematic objects is formed by the *geo-objects*, which are characterized by an attribute that is a geometric object. A geo-object therefore has a location and spatial extension. If a geo-object is complex, a number of integrity constraints need to be enforced. For example, the geometry of a complex geo-object has to be the union of its component geometries.

Geometric Objects

Associated with each geo-object is a *geometric object*, which is typically composed of elementary geometric objects. These elementary geometric objects form the class *Geometry* with its three subclasses *Region, Arc,* and *Point.* An arc may be curved although the current implementation is restricted to (piecewise linear) polylines. Between these classes there exist the usual geometric relationships: A region has several bounding arcs. An arc may inturn belong to any number (including zero) of regions. Similarly, an arc has two endpoints, and any point may be the endpoint of any number of arcs.

Note that a geometric object can either be a singular point, arc, or region, or a collection of such singulars. If these singulars all belong to the class *Region* then the resulting complex object belongs to the class *RegionSet*. The classes *ArcSet* and *PointSet* are defined analogously. However, if the collection of singulars is heterogeneous in the sense that it contains singulars of different types, it belongs to the class *GeometrySet*, which is the superclass of *RegionSet*, *ArcSet*, and *PointSet*.

This design guarantees that the result of a boolean set operation on two geometric objects can always be represented by exactly one geometric object. It also enables us to represent the geometries of any geo-object, however oddly shaped, with just one geometric object.

Graphic Objects

Graphic objects are used for the (interactive or printed) display and the interactive update of thematic objects, in particular of geo-objects. The looks of a graphic object are defined in detail by its attributes, such as color, line width, or text font. Possible graphic representations include business graphics, tables, videos, raster images, or GIS-typical vector graphics. A thematic object can be linked to several graphic representations (e.g., for multiple scale display).

An important subcategory of graphic objects is formed by the *cartographic objects*, which are used for the graphic display of geo-objects. With this design, GODOT implies a clear separation between geographic and cartographic information. Cartographic objects contain methods that determine how the properties of a given geo-object are represented in terms of the graphic available. In particular, questions of scale and cartographic generalization are handled at this level and not at the level of thematic objects.

2.2.3 Geographical Information System Entity Relationship (GISER)

- *Introduction*

The GISER model has described GIS as a collection of four major functional units:-

Functional Unit	Data Modelling Requirement	Example/Issues
Data Input	Data consistency and Quality	Constraints, Interpretation
	Continuous Space	Discretization, Interpolation models
Data Modelling	Geometry	Points, curves, polygons etc
	Topology	Networks, Partitions
Data Manipulation	Set Operations, Spatial Relationships, Network Analysis	Topological, Direction, Metric
Result Presentation	Visual Representation	Visualization constraints, Maps

Table 1: Summary of Requirements

1. Data Input Unit,
2. Data Model,
3. Data Manipulation Capabilities, and
4. Result Presentation Facilities.

Table 1 summarizes the modelling requirements of a GIS in relation to each of its functional units.

Data Model

The GISER model[8] is shown in Figure 4. It uses the enhanced Entity Relationship[24] diagram notation described in Figure 5, along with dashed lines for continuous fields and relationships. The GISER model is based on four major concepts: Space/Time, Features, Coverages and Spatial Objects. Space and time represent boundless multidimensional extents in which geographic phenomena and events can occur and have relative position and direction. GISER is a continuous field and may possibly be discretized into realms and calendars if needed. Examples include of realms the surface of the earth and its subsets of interest.

Features represent geographic phenomena such as rivers, vegetation, cities, etc.

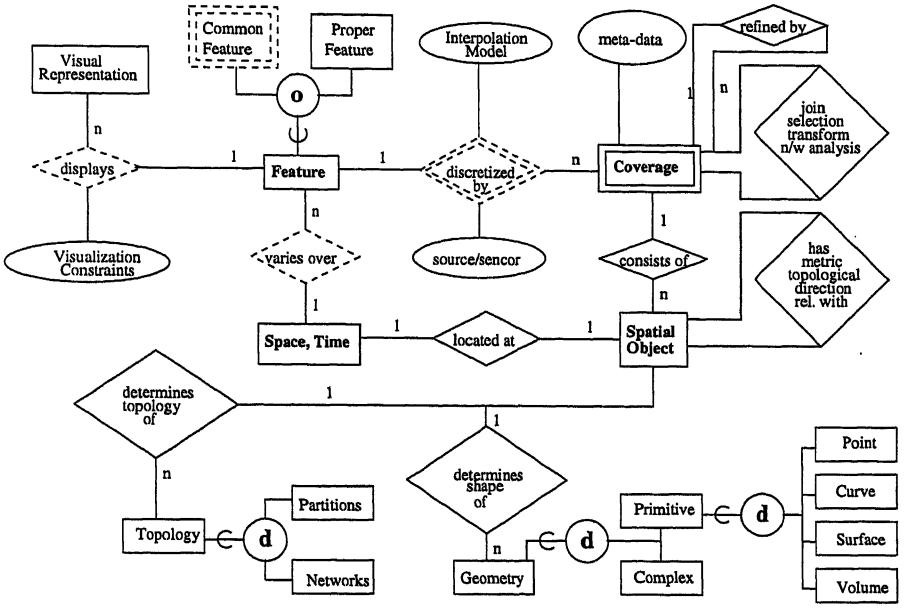


Figure 4: The GISER Model

GISER models features as continuous fields varying over space and time, but features as such cannot be stored in a GIS. These must go through the process of discretization in order to compute coverages which are then stored in a GIS.

A feature may have multiple coverages based on multiple discretization with varying resolution, accuracy and sources. A coverage consists of a set of spatial objects. A spatial object occupies a subset of space and time. It has geometric and/or topological properties. A geographic object consists of a spatial object and a non-spatial object. The spatial object represents the location and spatial extent of the object at some point in time. The non-spatial object represents other attributes such as its name.

Data Input Unit

The key data-modelling issues for the data-input unit relate to data quality and to the discretization of continuous space.

Data consistency and Quality: The measurement and discretization process is prone to errors and inaccuracies. Raw geographic measurements data is often processed and interpreted. It is processed to filter noise or out-of-range data. Then

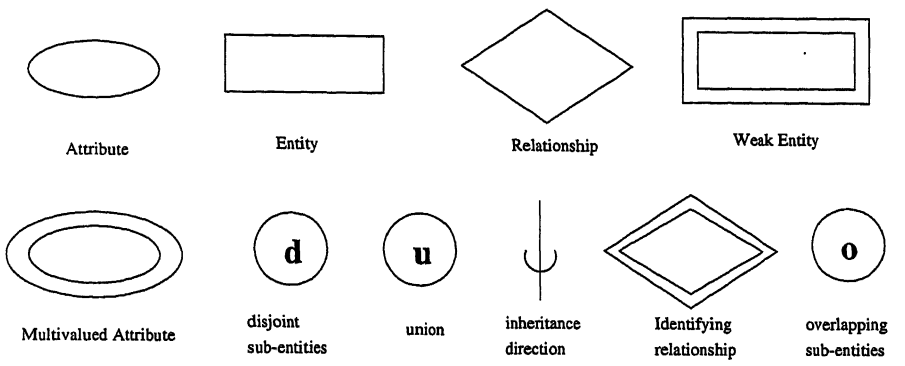


Figure 5: Enhanced Entity Relationship (EER) Model

it is interpreted to fit a domain prototype, and from that, outliners are removed. If such data is used in an analysis, the results of the analysis would need to be interpreted within the error tolerance of the data. Thus, accurate representation of data is critical for the proper use of geographic data in decision making[25].

Discretizing Continuous Space: Geographic phenomena occupy continuous space. However, computers and digital databases can only store and manipulate their discrete approximations. The process of discretization raises the issue of interpolation, which estimates the values at various points to generate a cartographic presentation. Triangulization-based discretization allows simple interpolation.

Discretization could either be uniform, i.e. independent of the spatial entity that it is modeling (e.g. laying a rectangular grid over a continuous 2-dimensional domain), or it could be dependent on it (e.g. a thematic layer[4]). A layer is a mapping from a domain of non-overlapping geometric regions to a domain of attribute values. In a simple case, a thematic layer consists of a set of polygons, with each polygon being an area of constant value for an attribute. Thematic polygons are weak entities. These entities do not exist, except to determine the values of an attribute.

The GISER model specifies the following characteristics for a data input unit of a GIS:

Data Consistency and Quality: GISER provides the relationship *refined by* to support the data collection process as well as multiple levels of refinements. GISER states that raw data is refined into processed data, which is refined into interpreted

data. Ancillary descriptive data, often called *meta-data*, is provided to facilitate the interpretation and use of the processed data. Meta-data on each coverage would document the refinement procedure and the source.

Continuous Space Modeling: GISER includes continuous fields named *features and space/time* as well as the relationship *discretized by* to support continuous space. Features represent the space varying characteristics of a GIS entity. Each feature represents a mapping from space to a domain of values. Examples of features include elevation, soil-type, and water-level. Some features are *Proper Features*, which represent a collection of uniquely named geographic places. Each instances of these entities has a unique name, and the entity's geographic location can sometimes change. Features which are not proper features are Common Features. A comon feature is identified by its location in space and time and does not have an identification of its own. These entities can be regarded as weak entities which are dependent on and identified by their spatial locations.

In order to accurately model these entities, GISER makes explicit the fact that continuous entities are discretized for representation in the database system. Features are discretized to give coverages. There could be multiple coverages for the same feature depending on the source/sensor. Different interpolation models need to be used for different coverages to retrieve a feature. Specifically, GISER includes the relationship *discretized by* with the attributes *interpolation model* and *source/sensor* to model this. A *coverage* has multiple *spatial objects*, and is modeled with the relationship *consists of*. A *spatial object* occupies a subset of space and time, and is modeled using the relationship *located at*. *Features* occupies a subset of space and time and its attribute takes values defined for a subset of space and time. This is modeled using the relationship *varies over*.

Data Model

Geographic entities have geometric and topological aspects.

- Geometry - Graphical entities have geometric properties which can be modeled by the measurements, properties and relationships of points, lines, angles, surfaces, etc. In GIS research and practise, two types of geometric data-types

are prevalent: *Vector data* and *Raster data.* Vector entities include points, lines, and polygons, all of which are representations of the space occupied by real-world entities. Raster data is characterized as an array of points, where each point represents the value of an attribute for a real-world area. Baumann[26] gives a more thorough description of various raster image types.

- Topology - Some geographic entities have topological properties which are unaltered by elastic deformations. Examples include the connectedness of a region and the connectivy between road intersections via road segments. Primitives are required for representing networks, graphs and partitions as high-level entities[11]. Partitions are related to networks in that they associate regions with other regions by relationships such as next, adjacent, etc. Networks and partitions can be decomposed into points, lines and regions for modeling; however, that complicates behavioral modeling (e.g. an operation such as the shortest-path computation) and manipulation. It is more natural to use direct construct for networks, partitions, etc.

GISER provides the following concepts to meet the data modeling requirements of a GIS:

- Geometry-In the GISER model, *Geometry* is an entity which is related to a spatial object by the relationship *determines shape of.* Additional entities represent the primitives such as points, lines, polygons, etc. as proposed in related models[13, 12].

- Topology- *Topology* is a property belonging to a spatial object and that property remains unaltered even when the object deforms. In order to represent the topology, the basic primitives such as networks (i.e. graphs) and partitions are provided. Additional primitives can be added on lines of the Worboy model[12].

Data Manipulation Capabilities

Queries in GISs often involve set operations on the geometric and topological

properties of spatial entities. These set operations could be classified into the following groups:

- Spatial Selection: A subset from an entity set that fulfills a spatial predicate.
- Spatial Join: A spatial join produces a set of pairs of spatial objects from two layers/entities that satisfies a spatial predicate.
- Transformation: A Transformation synthesizes a set of layers (a set of spatial objects) into a new layer using spatial predicates.
- Network Analysis: A network analysis represents a set of queries on spatial networks, such as route evaluation, network overlay and path optimization. A route evaluation is concerned with aggregating attribute data over route-units. A route-unit represents a collection of arcs with common charateristics (e.g. name). A network overlay enables the integration of disparate network-attribute databases, which join two or more sets of attributes.

Queries in GISs use spatial relationships within the query predicates. Spatial relationships can be distinguished into three categories[27]:

- Topological relationships: These include connected, adjacent, inside, and disjoint. These are invariable under topological transformations like translation, scaling, and rotation.
- Direction relationships: These include above, below, or north_of, southwest_of.
- Metric relationships: These include relationships such as the distance between two entities.

In GISER, spatial relationships are added to accomodate spatial operations in GIS. These relationships involve the location of the objects. Examples are north_of, south_of, northeast_of etc. *Topological relationships* involve the regions occupied by the objects. Examples are adjacent_to, inside, etc. *Metric relationships* could involve the geometry and location of the objects. Examples include the distance between two objects, or the area occupied by an object. To simplify the figure, only

the relationships involving *Spatial Object* are kept. A query in GIS is a set operation on one or more *coverages* to give another *coverages* to give another coverages. This set operation could be join, select, transform or analyze network.

Result Presentation Facilities

Result in GIS are visually represented in the form of maps, tables, plots, etc. Visual representation should be able to present all the required information and should not present any information which is not intended. Geographic data is most often displayed on a map. But information on a map is sometimes distorted to show some useful feature.

GISER proposes that *visual representation* to be specified in the database. It envisions this new database functionality as having the ability to communicate the essential properties of the display that the user requests, directly to the user interface. GISER adds the entity *Visual Representation* for this purpose. *Visual representation* consists of primitives such as text, icons, graphs and geometries like points, lines, polygons, etc. These primitives are associated with their location and orientation inside a visual representation. In addition, certain *visual constraints* are associated with the relationship *display*. These constraints ensure that *visual representation* does not convey any information which is not present in the geographic data, and that the visual representation should convey all the information requested by the user.

2.3 Conclusion

Mapget is a model of a map editing kernel as well as a general architecture for GDUIs. Its architecture has been used in a recent prototype[6], using on the one hand the O_2 DBMS[2] and the geographic layer GeO_2[10], and on the other hand the X11/Motif environment[14, 15].

The GISER data model integrates the field-based models of geographic data by using the *discretizes* relationship between feature fields and coverage entity. This leads to a simple data model for geographic data like the loop-detector dataset in

transportation, which is based on the concepts of space/time, features, coverages and spatial objects. Several data management aspects of data input, modeling, query and result presentation can be supported by the simple integrated model.

GODOT has a four layer architecture, consisting of a commercial object-oriented database system; an extensible kernel with classes and methods for representing complex spatial and non-spatial data objects; a collection of base components for query processing, graphics, database administration, and data exchange; and several user interface models. GISER is implemented using ObjectShore as database and X Window System for user interface.

Chapter 3

Global Positioning System

3.1 Introduction

In this chapter, we introduce the concept Global Positioning System (GPS) with particular emphasis to the GPSR 2000 receiver. The first section provides details of the GPS, the second section provides insight into the GPSR 2000 receiver and the final section is devoted to the GVISION software that manages the GPSR 2000 receiver.

3.2 The Global Positioning System

3.2.1 Introduction

The global positioning system (GPS) is a space based radio positioning, navigation and time transfer system which operates at all times of the day, under all weather conditions and everywhere on or near Earth.

GPS consists of three segments namely Space, Control and User. The space segment is the set of 24 satellites orbiting the earth once in 12 hours in six orbital planes with four satellites in each orbital plane. Control Segment based at four locations on the earth, monitors and controls the satellites. User Segment is the Global Positioning System Receivers (GPSR) used by a large number of users all over the world.

GPSR provides accurate and absolute position, velocity and time information which could be used for:-

- Intelligent Vehicle Highway System (IVHS),
- Personal navigation by Land, Sea and Air,
- Management of fleet taxis, buses and police vehicles,
- Land and construction Surveying,
- Mapping to create highly precise maps,
- Animal Tracking,
- Synchronization of time,
- Vehicle security systems,
- Communication sets with GPS for Police,
- Integrated navigation solutions.

GPS provides many advantages over other navigational systems such as Loran receivers (LOng RAnge Navigation)[28], Marine naviigational system[29], Inertial navigational system[30] etc. The decisive advantages are as follows:

- 24 hours world wide coverage,
- Independence from weather conditions,
- Precise 3 dimensional position and time information,
- Resistance to jamming,
- Easy integration/adaptability with other systems,
- Can be used on Land, Sea or Air.

3.2.2 Theory of Operation of GPS

The GPS system consists of three segments

- Space Segment
- Control Segment
- User Segment

Space Segment

Space Segment consists of a set of 24 satellites orbiting the earth once in 12 hours. The satellites are placed at an altitude of approximately 10.898 nautical miles above the surface of the earth and these satellites are arranged in six orbital planes inclined at 55° with four satellites in each orbit. Each satellite continually transmits the following signals.

- L1 signal at 1575.42 MHz
- L2 signal at 1227.60 MHz

The above frequencies are phase modulated by two ranging signals namely, P-code for military users and C/A code for civilian users. In addition, navigation messages are modulated and transmitted along with the ranging signals. Satellites use precise atomic clocks for synchronization with GPS time.

Control Segment

Control Segment consists of a master control station, four monitor stations located strategically around the earth to maximize satellite coverage and ground antennas. The monitor stations receive satellite data and communicate to the master control station. The master control station updates navigation messages to all satellites through the ground antennas.

- *User Segment*

User Segment consists of a large number of GPS receivers (GPSR) used all over the world. The GPS receiver, which is of current interest, receives and processes satellite signals to compute user's position, velocity and time.

The principal of computation of the user's position using the GPS satellite signals in a receiver can be described in short as follows. The GPS constellation of satellites is so designed that sufficient number of satellites are visible at any point on the surface of the earth at all times. Each of the set of 24 GPS satellites point on the surface of the earth at all times. Each of the set of 24 GPS satellites transmits a unique pseudo-random code continuously. GPS receiver uses correlation techniques to acquire satellites signals, extract navigation messages and perform ranging measurements. GPS receiver measures transit time of the signal and hence range from satellites to the receiver. Range measurements from four satellites are needed to locate precisely the user's position in the three dimensional space. Ranging measurements from three satellites are necessary to compute user's position in two dimensions.

3.2.3 Functional Components of a GPS Receiver

A typical GPS receiver used in civilian applications would have the following functional components:

- Antenna,
- RF Down Converter,
- Correlator, Navigation Module,
- User interface.

The remainder of this section provides details of a specific GPS receiver known as GPSR 2000.

GPSR 2000 Receiver

Accord's GPSR 2000 is a global Positioning System Receiver chipset solution designed around a programmable platform - Analog Devices' ADSP 2106X (SHARC) processor. SHARC is a 32 bit floating point DSP with on-chip SRAM and integrated I/O peripheral support. GPSR 2000 along with an RF front, a commercially available GPS antenna and keyboard/display unit forms a complete GPS receiver.

GPS 2000 is designed to simultaneously acquire and track 8 GPS satellite signals and compute user's accurate position, velocity and time. Accord's GVISION software provides a Graphical User Interface for the 2000 based GPS receiver on an IBM-PC which can be connected to the GPS receiver through an RS 232 link.

Functional Description

L1 GPS signal, received through an active/passive antenna, gets down converted to a low IF in the RF front end - designed around a VLSI down converter. This IF signal is sampled at a predefined rate and then quantized in an A/D converter inside the RF front end. The quantized samples of the IF signal are received serially by the Digital Signal Processor (DSP) and stored in an internal buffer. The down conversion of this stored IF signal to base band, correlation of the signal with the local code, carrier and code tracking loops and computation of user's position, velocity and time as well as the communication interface to a host are all implemented in software running on the DSP.

Like other receivers, the GPSR 2000 receiver consists of:-

- Antenna section - An antenna section consists of:-
  - An antenna operating at 1575.42 MHz (L1 Signal)
  - A built-in low noise amplifier (LNA)
  - Coaxial cable with SMA connectors connecting antenna/LNA to the RF Front End
- RF Front End - RF Front end is designed around a commercially available VLSI RF down converter with a set of RF passive components. It receives

the input signal from the antenna section and provides down converted digitized IF signal to the correlator. RF front end consists of the following major components:

- Bandpass filter @ 1575.42 MHz
- VLSI Down Converter
- LC filter
- SAW filter
- Reference clock

The main functions of the RF Front End are:

- Reception of L-band GPS signal from the antenna
- RF signal amplification and filtering
- Down conversion of RF signal to a low IF in three mixer stages
- Digitization of the low IF output signal
- C/A code compatibity
- Built in test facility
- On chip PLL including VCO

- Correlator - Correlator receives digitized low IF output from the RF Front End and performs measurements on the satellite signal. The measurement outputs are sent to the navigation module which computes the user's position, velocity and time. The correlator consists of:
  - Buffers to store sampled low IF signal
  - Carrier generator
  - Code generator
  - Code correlator

  The main functions of the correlator are:

- Reception and storage of digitized low IF signal in buffers,
- Down-conversion of low IF signal to baseband,
- Parallel processing of eight channels,
- Generation of the local carrier,
- Generation of local C/A codes for 24 satellites,
- Correlation of promt and dither samples of incoming code
- Interface to Navigation Module

- Navigation Module - navigation Module receives measurements performed on the satellite signal by the correlator. The pseudo range and delta range measurements for the satellites being tracked by the correlator are used by the Navigation Module to estimate the user's position and other parameters The navigation module consists of the following modules:
  - Correlator manager,
  - Satellite database manager,
  - Satellite selection,
  - Channel manager,
  - Position solution,
  - Interface to Correlator and user interface,
  - Real time multitasking executive,
  - Standard interface through RS-232,
  - Measurement data processor.

- User Interface - User interface software supplied along with the GPSR 2000 receiver known as GVISION runs on an IBM-PC connected to the receiver on an RS 232C link. For details, see Appendix A. Software has the following facility:-
  - entering the estimate of user's position(langitude, longitude and altitude).

- entering the estimate of Time
- Selection of Geodetic Datum
- Selection of Cold, Warm or Hot start
- Monitor satellites being searched, tracked
- Monitor the Receiver status information
- Display of computed user's position, velocity and time
- Upload/download of almanac data
- Selection of 2D or 3D position mode.

3.3 Relation between GPS and GIS

GPS is one of the essential elements of a GIS. GPS receiver collects the data in the form of images or co-ordinates and provide specialized capabilities for manipulating, analyzing, and visualizing those data. GPS provides this data as an input to the GIS, but lack of strong geographic data management and analytical operations, sets it apart from GIS technology.

Chapter 4

System Design Considerations for GIS

4.1 Introduction

This chapter is devoted to the description of the proposed design of the GIS model. In the first section, we discuss the general features that have to be provided in the GIS model, the second section describes the architecture as well as the detailed description of the components of the system, the third section provides the detail of the GIS application built on the proposed model and the final section state the assumptions made while designing the system.

4.2 Requirements

As explained in Chapter 2, GIS is a system for capturing, storing, checking, integrating, manipulating, analyzing and displaying data which are spatially referenced to the Earth. Key components of GIS are:-

- Tools for the input and manipulation of geographic information
- A database management system (DBMS)
- Tools that support geographic query, analysis, and visualization

- A graphical user interface (GUI) for easy access to tools

Rest of this section presents various requirements of a GIS in relation to each of its functional units.

4.2.1 Input and Manipulation tool

GIS aquire data from GPS, remote sensing, field data and maps already stored in some other databases. The data is first converted to a suitable digital format before using it in GIS. The process of converting data from paper maps into computer files is called digitizing.

Modern GIS technology can automate this process fully for large projects using scanning technology; smaller jobs may require some manual digitizing (using a digitizing table). Today many types of geographic data already exist in GIS-compatible formats. These data can be obtained from data suppliers and loaded directly into a GIS.

It is likely that data types required for a particular GIS application will need to be transformed or manipulated in some way to make them compatible with the system.

4.2.2 Database Management System

For small GIS projects it may be sufficient to store geographic information as simple files. However, when data volumes become large and the number of data users becomes more, then a database management system (DBMS)[31] is required to help store, organize, and manage data at various locations.

There are many different designs of DBMSs[31], but in GIS the relational design has been the most useful. In the relational design[32], data are stored conceptually as a collection of tables[32]. Common fields in different tables are used to link them together.

4.2.3 Query and Analysis tool

Geographic data is queried and analyzed for various operations. After receiving query from user, query and analysis tool either interacts with database or visualization tool. Operations of this tool can be categorized as:-

- Map display operations,
- Manipulating and updating maps,
- Interaction with database.

Given below are the examples of various end user operations of query and analysis tool:-

- *Map display operations*
  - Display a map.
  - Display only a portion of the map.
  - Allow user to change scale on a whole map or to a subset of a map.
  - Merge two maps.
- *Map manipulation and updation operations*
  - Update the objects, either their geometry or their description.
- *Interaction with database*
  - Getting the objects from the database, to be displayed.
  - Upadating objects.
  - Attaching a representation to a geographic object.
  - Querying maps. Queries can be classified into two categories:-

1. Alphanumeric queries - These queries require interaction with the description. An example of such a queries are:-
   - Showing description corresponding to any object.
   - Search for some attribute in the description.
2. Graphical queries - These queries require graphical input. For solving such queries, system should be capable of processing images. Example of these queries are:
   - Distancing finding between two objects.
   - Area calculation.
   - Getting the nearest object of a given object.
   - Perimeter calculation.
   - Finding shortest path between two objects.

Visualization tool

For many types of geographic operation the end result is best visualized as a map or graph. Maps are very efficient at storing and communicating geographic information. GIS provides new and exciting tools to extend the art and science of cartography. Map displays can be integrated with reports, three-dimensional views, photographic images, and other output such as multimedia.

Graphical User Interface

As it is common in geographic applications, the layout of an application interface consists of a set of windows with menus. These windows can be composed of other windows having semantic connections. Their contents are linked to the geographic database. The GUI is then more than a simple display module, and it requires in particular interaction with the user for:-

- Selecting objects in windows.
- Selecting area for zooming.

- Modifying the aspects of cartographic objects interactively.
- Formulating geographic queries, either visually or through a language such as SQL.
- Providing at higher level:
  - Tools for browsing the database and the map library to select appropriate maps for a given study.
  - A help environment.

Given these requirements, it becomes clear that the interface has to communicate with the database for:-

- Retrieving the objects to be displayed. These objects may be maps as a whole i.e., the geometric part as well as their description.
- Attaching a representation to the geographic object.
- Updating objects.
- Querying maps.

4.3 Architecture

Figure 6 shows the architecture of the proposed design. It consists of following components:-

- GUI,
- Query and Analysis tool,
- Visualization tool,
- Input and Manipulation tool and
- DBMS.

The rest of this section provides the description of the individual components of the system.

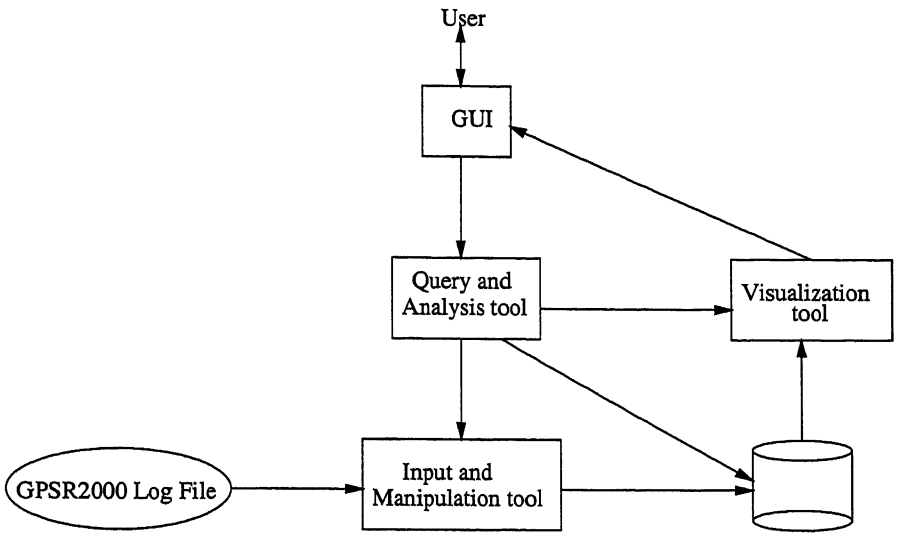


Figure 6: Functional Architecture of Proposed Model

4.3.1 GUI

GUI is the component with which a user directly interacts. Whenever it receives a request from user, it passes it to the Query and Analysis tool to service the request. GUI consists of windows, menus and buttons, which are selected by the user. For details, see Appendix B.

4.3.2 Query and Analysis tool

Query and Analysis tool receive requests from GUI. After receiving request from GUI, it decides whether request should be passed to Visualization tool or to DBMS. Requests which require interaction with database are passed to DBMS e.g., updating an object, entering information corresponding to an object, e.t.c. Requests which require translation of geometry of map as represented in the database into some other structure for display, are passed to Visualization tool e.g., Changing the scale of the map, e.t.c. User requests for reading the input from position trail file are transferred to Input and Manipulation tool.

4.3.3 DBMS

DBMS receives request from Query and Analysis tool, process it and sends the output to Visualization tool. Underlying DBMS can be distributed. Local database stores the location information corresponding to each geographic-object. Maps are the basic objects stored in the database and to be manipulated by the end user. A map is a collection of geographic objects. A geographic object is usually derived from an entity of the real world. It has a spatial part as well as a (alphanumeric) description, i.e.,

```
map = {geographic-object}
geographic-object = (description, geometry)

A geographic object can be composed of several other geographic
objects, i.e.,

geographic-object = (description, geometry)
                    /* atomic geographic object */
                  | (description, {geographic-object})
```

This representation permits to associate a geometry with only atomic objects. Geometry of complex objects can be inferred using propogation mechanism[33] from the geometry of atomic objects that compose them with a possible recursion.

The description part of the geographic-object stores the location information(URL). This information pertains to the physical location of the server on which the detailed information of the geographic-object is stored and the full path name, where this geographic object can be found on its server.

4.3.4 Visualization tool

Visualization tool maps the three-dimensional geographical information in the form of grid map, with each co-ordinate having potentially additional attributes. For an effective visualization tool in the targeted application, we assume that the variation in the third dimension (height) is small compared to the extent of x and y ranges.

Visualization tool receives request from Query and Analysis tool or DBMS. Initially users want to display a map identified by a name. Once it has been displayed, they might ask for the description of the displayed geographic objects. It is the responsibility of Visualization tool to convey the output of every request to the user.

4.3.5 Input and Manipulation tool

User can invoke the system by specifying the name of the map. By default, system reads the position trail file saved by GVISION and draws its map in the main window. For entering other attributes in the database, user sends the request via GUI, which in turn sends it to Input and Manipulation tool to save the information in the database.

4.3.6 Design Approach

GIS integrates GUI and DBMS. There are possible two approaches of integration: either by strong integration[16] or by weak integration[16]. We are in favor of the second alternative because of modularity and openness. A very promising solution is to define a generic graphic interface independently of the DBMS, and to provide conceptual tools for communicating and for mapping both environments. From a design point of view, there is a total independence between the database and the interface. Hence new tools can easily be added in future.

4.4 IITKLAN-GIS: A GIS Application

IITKLAN-GIS is a GIS application, based on the proposed model, over a local area network. Local area network consists of three-segmented fibre optic cable, which connects every department in the campus. IITKLAN-GIS generates a two-dimensional grid map of the LAN, with the assumption that the detailed network information of the individual department is located at the corresponding department server. Hence for accessing detail information of the department, local machine

communicates with the remote department server.

Currently, IITKLAN-GIS provides the GIS features like calculating the distance between two nodes of the map, finding the nearest node etc., but in future it can be enhanced by merging with SNMP.

4.5 Assumptions

Following assumptions have been made while designing the prototype:-

- Input and Manipulation tool reads the position trail file of the GVISION software for input and this file contains the co-ordinate information of each object in the map.
- Once a map is created, there is no deletion or addition of object in the map.
- At a time, only one map can be edited and shown in the window. It is not possible to edit only a part of the map.
- Database stored at various locations are assumed to be having same format.

Chapter 5

Implementation Details of a GIS for Network Monitoring

5.1 Introduction

This chapter presents the description of a prototype kernel for GIS. In the first section, we describe the implementation details of this kernel which includes system details, protocols and operating environment. In the second section, we explain the data structure and search algorithm used for the implementation of GIS. In the last section, we detail its features.

5.2 Prototype Implementation

GIS can be thought of as an information system consisting of two environments. On the one hand, graphic tools for displaying maps, and on the other hand, a (geographic) database. GIS communicates with both environments and is composed of three tools: Query and Analysis tool, Visualization tool and Input and Manipulation tool. Visualization tool is in charge of displaying maps in windows. Query and Analysis is in charge of categorizing the queries and Input and Manipulation tool is in charge of addition of maps into the database.

5.2.1 Operating Process

Access to the GIS is through a GUI. The GUI passes the request to Query and Analysis tool. This request is of the form:

```
<command, options-list>
```

Query and Analysis tool process the command and checks whether the request is for map displaying, map updation, database querying or map addition. For providing map display service, Query and Analysis tool transfers the request to Visualization tool. Visualization tool executes this command and transfers the output map to GUI. For providing map updation and database query services, Query and Analysis tool transfers the request to DBMS. DBMS executes this command and sends the result to Visualization tool, which in turn sends it to GUI.

The addition of a map to the DBMS also takes place through Query and Analysis tool. The map read from the GPSR 2000 Log file is first converted into a set of geometric descriptions by Input and Manipulation tool. These geometric descriptions are then stored in the DBMS. At the same time, information pertaining to each object of the map are inserted into the DBMS through Query and Analysis tool.

5.2.2 Operating Environment

The prototype has been implemented using Java language[34] and UNIX filesystem. GUI, Query and Analysis tool, Visualization tool and Input and Manipulation tool are implemented using Java while database information of DBMS is stored in UNIX filesystem. This database information is distributed, hence DBMS is also responsible for accessing remote information. Java is an object-oriented language with the following features:-

- Simple: Java is designed to be easy to learn and use effectively.
- Secure: Java is a highly secured language as it confines its program to the Java execution environment and not allowing it access to other parts of the computer.

- Portable: Java can produce secure platform-independent applications that can come from anywhere on the internet. Java programs are interpreted rather than compiled, it is much easier to run them in a wide variety of environments. Only Java run-time system needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it.

- Multithreaded: Java is designed to meet the real-world requirement of creating interactive, networked programs. To accomplish this, Java supports multithreaded programming, which allows one to write programs that do many things at the same time.

- Dynamic: Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. This makes it possible to dynamically link code in a safe and expedient manner.

- Interpreted and High Performance: Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java bytecode. This code can be interpreted on any system that provides a Java run time. Unlike other interpreted systems (e.g., BASIC, Tcl, and PERL), java bytecode is carefully designed so that it would be easy to translate directly into native machine code for very high performance.

- Robust: Java executes programs on a variety of systems. To gain reliability, Java restricts user in a few key areas to find mistakes early in program development. Also Java frees user from having to worry about many of the most common causes of programming errors as it manages memory management itself and provide object-oriented exception handling for exceptional conditions. Java is a strictly typed language, it checks the code at compile time. It also checks the code at run time.

- Distributed: Java is designed for the distributed environment of the internet because it handles TCP/IP protocols. *Remote method invocation (RMI)* package provided by Java, brings an unparallel level of abstraction to client/server programming.

- Support for Graphics: The Abstract Window Toolkit provided with Java contains numerous classes and methods that offers a programmer many tools to design a graphic user interface. It can handle text as well as pictures and are well suited for the design of geographic interfaces. The AWT supports windows, fonts, various controls, such as scroll bars, menus, popup windows, and push buttons e.t.c., which allows programmer to create an user friendly graphical interface.

5.3 Data Structure and Search Algorithm

Organization of data is an important factor for deciding the performance of a GIS package. Database management techniques range from simple spreadsheets[28] to complex data structures. Relational databases[32] and object-oriented methods[35] are very different in their internal structure, but both methods have proven successful in similar applications.

We investigate how data structures can support efficient access to data, this includes the investigation of the suitability of classical data structures (such as search trees or hashing techniques)[36] for the GIS application. GIS deals with very large volumes of data, such system require very fast response to the queries.

We design data structure for our GIS package, so that it retrieves information efficiently. Main assumption is that GIS database is very large and once database is created, addition and deletion are less frequent than the updation and search operations.

Also, the node name entry in every record is unique. In our application, retrieval and updation are more frequent than addition and deletion, hence search time should be optimum, to overall increase the efficiency of the GIS package. For searching, whole table is loaded into the primary memory where each record contains

information corresponding to individual node. Node name entry in every record is unique and hence is a key field to differentiate among different records.

Our search algorithm accepts an argument a and tries to find a record whose key is a. The algorithm may return the entire record or, more commonly, it may return a pointer to that record. It is possible that the search for a particular argument in a table is unsuccessful; that is, there is no record in the table with that argument as its key. In such a case the algorithm may return a special "null record".

The simplest form of a search is the sequential search. The algorithm for sequential searching examines each key in turn, upon finding one that matches the search argument, its index is returned. If no match is found, -1 is returned. Sequential searching is not very efficient. For searching through a table of constant size n, the number of comparisons depends on where the record with the argument key appears in the table. If the record is the first one in the table, only one comparison is performed, if the record is the last one in the table, n comparisons are necessary. If it is equally likely for the argument to appear at any given table position, a successful search will take on the average $(n+1)/2$ comparisons, and an unsuccessful search will take n compariosns. The number of comparisons is $O(n)$.

If the table is stored in ascending or descending order of the record keys, there are several techniques that can be used to improve the efficiency of searching. This is true if the table is of fixed size. One obvious advantage in searching a sorted file over searching an unsorted file is in the case that the argument key is absent from the file. In the case of an unsorted file, n comparisons are needed to detect this fact. In the case of a sorted file, only $n/2$ comparisons on the average are needed.

The most efficient method of searching a sequential table without the use of auxillary indices or tables is the binary search. The argument is compared with the key of the middle element of the table. If they are equal, the search ends successfully, otherwise either the upper or lower half of the table must be searched in a similar manner.

Each comparison in the binary search reduces the number of possible candidates by a factor of 2. Thus, the maximum number of key comparisons is approximately $\log_2 n$. Actually it is $2 \times \log_2 n$. Binary search is appropriate for the application

where search is very frequent than deletion and insertion. Deletion create holes in the table, therefore after every deletion table should be rearranged. Insertion is quite expensive in sorted table, as every record should be inserted at its proper location. For this, locate the actual location of the record in the table, shift forward by one position all the following elements up to the first empty position to make room for the new element. But in our GIS application, deletion are rare and insertion can be done off-line to increase the overall performance of the package.

5.4 Features

The End users are expected to query and edit maps by means of various operations. These operations are implemented in various steps. Our primary aim is to implement these operations transparently. We describe below the various functionalities supported by our prototype:-

- Display - The display of a map is implemented by function *Display(m, w)*. Display consists of the following steps:
  - Create window w instance of class Frame.
  - Read a map m in the database and visualize the map in the corresponding window w.
- Zoom - Zooming is an end user request that takes as parameter:-
  - A scale factor f, defining the magnifying factor to be applied to the displayed map.

  The zoom operation consists of following steps:
  - Magnification of the geometric part of all objects contained in the map, according to the scale factor parameter f.
  - Display of the new map in a new window.
- Updation - The updation of an object in a map is implemented by function *Update(m, o1, o2)*. This consists of following steps:

- Fetch object *o1* from map *m* and replace it with object *o2*.
- Store the fetched object back into database.

- Search - Search function is implemented as *Search(m, o)*. It consists of following steps:-
  - Bring the objects of map *m*.
  - Search for object *o*.
  - Display the output.

- Distance - Distance between the two objects of a map is calculated by the function *Display(o1, o2, m)*. It consists of following steps:
  - Fetch the co-ordinates of the objects *o1* and object *o2* of the map *m* from database.
  - Calculate the distance between the two given objects.
  - Display the ouput.

- Nearest object - Function *Nearest(o1, m)* finds out the nearest object of the given object *o1* of the map *m*. It consists of following steps:-
  - Fetch all the objects of the map *m*.
  - Calculate the distances of all objects from the object *o1*.
  - Calculate the minimum distance.
  - Display the object with minimum distance.

- Show_Info - Function *Show_Info(m)* shows the detailed information of all the objects of the map *m*. Step involved in performing the function *Show_Info(m)* are:-
  - fetch all the objects of the map *m*.
  - Display the details of all the objects.

- Enter_Info - Function *Enter_Info(o, des, m)* enters the description-list *des* corresponding to each object of the object-list *o* of the map *m*. It consists of following steps:-
  - Create a map *m* into the database.
  - Enter the object-list *o* and description-list *des* into the database.
- Overlay - Function *Overlay(mlist, w)* displays each map of mlist. It consists of following steps:-
  - Create window *w* instance of class Frame.
  - Read each map of *mlist* from the database and visualize them in the window *w*.

Chapter 6

Summary and Conclusions

This thesis has concentrated on the problem of designing a Geographical Information System (GIS) and its implementation. We first studied various GIS model like Mapget, Godot and GISER. One important issue in the GIS model is the connection problem. We saw that this connection problem is related to the more general problem of connecting an external module to a DBMS. This can be solved in two different ways: Either by strong integration, which means that there is a single homogeneous system to consider and that everything is done according to the DBMS features, or by weak integration.

We also studied the Global Positioning System (GPS) with particular emphasis to the GPSR 2000 receiver, which provides input to our GIS model. We then proposed our model of GIS, which consists of five componenets: (i) GUI, (ii) Input and Manipulation tool, (iii) Query and Analysis tool, (iv) DBMS and (v) Visualization tool.

We then described our GIS implementation based on the proposed architecture. It uses Java for the implementation of GUI, Query and Analysis tool, Visualization tool, and Input and Manipulation tool and UNIX filesystem for storing information in the database.

6.1 Directions for Future Work

The implementation may be extended and refined in the following ways:-

- More graphical features like merging and clipping of maps can be added to the current GIS.
- Currenlty UNIX filesystem has been used for storing information, this can be replaced by some advanced database system.
- Image processing capability can be added to support complex queries.
- Input and Manipulation tool can be enhanced to understand various other formats.
- In this work, we have introduced some elementary queries, future work includes implementation of complex queries.
- Current work can be extended to handle a more complete set of user queries to the database.

Appendix A

GVISION

A.1 File Management in GVISION

GVISION uses the following files :

- Jump start data(jstart.dat) - This file is created after saving the current settings before exiting.
- default almanac file(default.alm) - This file is copied to GVISION directory at the time of installation.

GVISION creates the following files :-

- Position trail file (record.pos) - GVISION continuously stores Latitude, Longitude into this file from the instance the receiver has obtained GPS. This file is used to display the position trail. Every time GVISION is invoked, the previous record.pos is stored sa oldrecrd.pos and a new record.pos is opened. This file is used by GIS, for extracting the location information.
- Message log file(genmsg.log) - GVISION continuously logs messages coming from the receiver into this file. If genmsg.log file size increases beyond 1 MB, GVISION renames genmsg.log file as genmsg.bak and opens a fresh genmsg.log file.

- Almanac data file(<filename>.alm) - Almanac data can be loaded into these files. These files can also be used to send almanac to the receiver.
- Waypoints data(waypts.dat) - GVISION stores waypoints into this file.

A.2 GVISION Software

GVISION[37] is Accord's PC based interface for the GPSR 2000 Chipset based GPS receiver. GVISION allows the user to operate the receiver and use it as a location and navigation aid. Besides, GVISION can be used as a powerful evaluation tool for the GPSR 2000 Chipset based receiver.

A.2.1 Features of GVISION

GVISION has following features :-

- Warm start by allowing user to initialise the receiver with time, date, almanac and initial position estimate
- Upload/download facility for almanac
- Real-time display of user's position, time, date, velocity etc.
- Facility to reset the receiver,
- Connection integrity status between the PC and the Receiver.

A.2.2 User's Guide for the GPSR 2000 Receiver

Initialisation for the Receiver

The receiver can be initialized in three modes:

- AUTONOMOUS COLD START
- COLD START

- NORMAL START(WARM START)

When the GPS Receiver is switched ON, it needs an initial estimate of its position, GMT, current date and satellite almanac information to know which satellites to search for. If none of these data are provided to the receiver, the receiver is in AUTONOMOUS COLD START mode. If all these data are provided, the receiver is in NORMAL START mode. If only some of these data is provided, the receiver is in COLD START, the receiver tries to acquire signals from all the possible 24 GPS satellites, attempting eight at a time, because satellite visibility information is not available. During NORMAL START, the receiver computes satellite visibility information from the estimates supplied by the user and attempts to acquire signals from the visible satellites. The computed visibility information will be as precise as the estimates supplied.

When receiver is initialised, it will determine which among the 24 GPS satellites are visible and acquires signals from them. After collecting ephemeris data, range and doppler measurements from at least three satellites, the receiver will give continuous periodic position, velocity and time outputs, as long as it is able to get signals from at least three satellites which have a good geometry with the receiver.

∎ *Communication with the Receiver*

The receiver can be connected to either COM1 or COM2 serial ports of the PC. The port can be specified to GVISION either at the command line or by selecting Settings/Communication menu.

∎ *Position Solution Mode*

The receiver needs to track at least three healthy satellites and collect ephemris, range and doppler measurements from them to be able to compute its 2D position. Besides, the relative geometry of the user and the receiver should be such that the DOP is within acceptable limits. Similarly, it takes four satellites to be able to compute 3D position. Usually, the receiver automatically switches between 2D and 3D modes of position solution dynamically based on the number of satellites

being tracked, availability of ephemeris, range and doppler measurement data, DOP associated with the satellites etc. This is AUTO 3D mode which prevails in the receiver by default.

The receiver can be forced to compute only 2 Dimensional psition with an assumed constant altitude. This is called the 2D Solution Mode.

Communication Protocol

The GPSR 2000 receiver communicates with the host through a message protocol, Accord's custom binary format. This consists of three kinds of messages

- Receive messages - Receiver receives these messages from the host.
- Transmit messages - Receiver transmits these messages to the host.
- Bi-Directional messages - Receiver sends and receives to/from the host.
- Periodic messages - Transmit messages which the receiver sends periodically at specified time interval the host. These time intervals can be configured by the host. These messages switched ON and OFF by the host.
- Aperiodic messages - Transmit messages which are transmitted by the receiver only when requested by host.
- Immune messages - Periodic Transmit messages which cannot by switches OFF by the host.

System Requirement

Processor	:	80386 or Higher
Memory	:	1MB RAM
Video	:	VGA adapter with Color/Monochrome monitor
Disk	:	Atleast 2 MB free working space
Serial port	:	1

Appendix B

GUI

B.1 Options of GUI

GUI can be invoked by the command "java gis [filename]", where filename is the name of the database to be invoked. By default, GUI reads the position trail file (record.pos) of the GPSR 2000. When GUI is invoked without specifying any database, it shows the following options:-

- Save - This option allows the user to create a new database and save the contents of position trail file in the database. Apart from co-ordinate information, user can also save the description of these geographic co-ordinates into the database.
- Open - This option allows the user to open a new database.
- Zoom - This option magnifies the current map in the GUI window, according to the scale factor.
- Overlay - This option displays one or more maps in the GUI window.
- Exit - This option exits the user from the application.

When database is specified, GUI shows the following options:-

- Update - This option allows the user to update the database.

- Open - This option allows the user to open a new database.
- Show_Info - Show_Info allows the user to access the database.
- Search - This option allows the user to search the database.
- Enter_Info - Information related to the description portion of the geographic objects can be added using this option.
- Distance - This option allows the user to find the distance between two geographic objects.
- Nearest_Node - This option allows the user to find the nearest geographic object of the specified geographic object .
- Exit - This option exits the user from the application.

For details see Section 5.4. Clicking mouse button on any geographic-object (filled squares) on a map will bring up the detailed map of the corresponding geographic-object on the computer screen.

Bibliography

[1] A. Voisard. "Designing and Integrating User Interfaces of Geographic Database Applications". *Advances in Spatial Databases*, LNCS No. 653, Springer-Verlag, Berlin, pp. 23–33.

[2] M. Scholl and A. Voisard. "Object-Oriented Database Systems for Geographic Applications: An Experiment With O_2". Chapter 28 of "Building an Object-Oriented Database Systems: The Story of O_2", (1992).

[3] B. Amann, V. Christophides and M. Scholl. "Integrating Hypermedia Systems with Object-Oriented Database Systems". *In Proc. of the DEXA Intl. Conf.*, (1993).

[4] Nectaria Tryfona and Thanasis Hadzilacos. "Geographic Applications Development: Models and Tools for the Conceptual Level". *In Proceedings of the International Conference on ACM Geographical Information System*, 1995.

[5] L. Haas and W. Cody. "Exploiting Extensible DBMS in Integrated Geographic Information Systems". *In Advances in Spatial Databases (SSD'91)*, O. Gunther and H. J. Schek, Eds., LNCS No. 525, Springer-Verlag, Berlin. Lecture Notes in Computer Science No. 409, Springer-Verlag, 1989.

[6] A. Voisard. "Towards a Toolbox for Geographic User Interfaces". *In Advances in Spatial Databases*, LNCS No. 525, Springer-Verlag, Berlin, pp. 75–97.

[7] O. Günther and W. F. Riekert. "The Design of GODOT: An Object-Oriented geographic Information System". *IEEE Data Engineering Bulletin, 16(3):4-9, 1993.*

[8] Shashi Shekhar, Mark Coyle, Brajesh Goyal, Duen-Ren Liu and Shyamsundar Sarkar, "Experiences with Data Models in Geographic Information Systems". *Advances in Spatial Databases*, LNCS No. 525, Springer-Verlag, Berlin, pp. 52–61.

[9] V. Dalis, Th. Hadzilacos and N. Tryfona. An Introduction to Layer Algebra. *Technical Report CTI-94.01.2*, Computer Technology Institute, University of Patras, Greece, 1994.

[10] D. Benoit, R. Laurent and S. Guylaine. "GeO_2; Why Objects in a Geographical DBMS ?". *In D. Abel and B. C. ooi (Eds.), Advances in Spatial Databases 3rd Symposium, SSD'93*, Springer-Verlag, Berlin, 1993.

[11] R. H. Guting. "GraphDB: Modeling and Querying Graphs in Databases". *Proc. of Intl Conference on Very Large Data Bases*, 1994.

[12] M. F. Worboy, K. T. Mason and B. R. P. Dawson. "The Object-Based Paradigm for a Geographic Database System: Modeling, Design, and Implementation Issues". chapter 10, pp. 91–102., John Wiley & Sons, 1993.

[13] Draft Base Document 94-025R1., The Open Geodata Interoperability Specification, October 1994.

[14] R. W. Scheifler and J. Gettys. The X Window System. *ACM Transactions on Graphics*, 5(2):79–109, 1986.

[15] OSF. Motif 1.0 programmer's guide. *OSF Journal*, 1989.

[16] P. Rigaux, M. Scholl and A. Voisard. "A Map Editing Kernel Implementation: Application to Multiple Scale Display". *In Spatial Information Theory (COSIT'93)*, A. Frank and I. Campari, Eds., LNCS No. 716, Springer-Verlag, Berlin.

[17] M. Scholl and A. Voisard. "Thematic map Modeling". *In Design and Implementation of Large Spatial Databases(SSD'89)*, pp. 176–192.. Lecture Notes in Computer Science No. 409, Springer-Verlag, 1989.

[18] France Bull. GoPATH 1.2.0, Documentation Volume of the Public Release, 1993.

[19] ILOG. Ilog Views Version 1.1, Reference Manual, 1993.

[20] M. Stonebraker, J. Frew and J. Dozier. The Sequoia 2000 Project. *In Advances in Spatial Databases (SSD'93)*, D. Abel and B. C. Ooi, Eds., LNCS No. 692, Springer-Verlag, Berlin.

[21] M. Stonebraker and L. A. Rowe. "The Design of Postgres". *In Proc. ACM SIGACT-SIGMOD*, pp. 340–355, (1986).

[22] C. Lamb, G. Landis, J. Orenstein and D. Weinreb. "The ObjectStore Database System". *Communications of the ACM 34*, 10(1991).

[23] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier and S. Zdonik. "The Object-Oriented Database System Manifesto". *In Proc. First Int. Conf. on Deductive and Object-Oriented Databases*, Kyoto, 1989.

[24] Shamkant B. Navathe. "Evolution of Data Modeling for Databases". *Communications of the ACM 35(9)*, September 1992.

[25] C. R. Elshchlager and M. F. Goodchild. "Dealing with Uncertainty in Categorical Coverage maps". *In Proceedings of the International Conference on ACM Geographical Information Workshop*, 1994.

[26] Peter Baumann. "Management of Multidimensional Discrete Data". *Very Large Databases Journal*, 3(4):401–444, October 1994.

[27] R. Guting. "An Introduction to Spatial Database Systems". *Very Large Databases Journal*, 3(4), 1994.

[28] Long Range Navigational Systems. "Geographic Information System Loran- C Coverage Modeling". *The Proceedings of the Twenty-Second Annual Technical Symposium*, (October 18–21, 1993, Santa Barbara, CA). Bedford, MA: The Wild Goose Association. 1994.

[29] Marine Navigational Systems. http://www.kvh.com.

[30] Inertial Navigational Systems. http://www.tspi.elan.af.mil/ins.html.

[31] A. Voisard. "Geographic Databases: From the Data Model to the User Interface". Ph.D. thesis, University of Paris at Orsay, 1992.

[32] Ramez Elmasri and Shamkant B. Navathe. "Fundamentals of Database Systems". The Benjamin/Cummings Publishing Company, Inc., Second Edition, 1994.

[33] M. Egenhofer and A. Frank. "Object Oriented Modeling in GIS: Inheritance and Propogation". *In Proc. Auto-Carto 9*, 1989.

[34] Patrick Naughton and Herbert Schildt. The Complete Reference JAVA. Tata McGraw-Hill Publishing Company Limited, 1997,

[35] "Experiences with Object Data Models in geographic Information Systems". http://www.cs.umn.edu/research/shashi-group/abstract/cacm.abs.html.

[36] A. M. Tenenbaum, Y. Langsam and M. J. Augenstein. "Data structures using C". Prentice Hall of India, Fourth Printing, 1990.

[37] GPSR 2000 User's Guide.